PHP is a general-purpose scripting language geared toward web development.[5] It was originally created by Danish-Canadian programmer Rasmus Lerdorf in 1994.[6] The PHP reference implementation is now produced by The PHP Group.[7] PHP originally stood for ***Personal Home Page***,[6] but it now stands for the recursive initialism ***PHP: Hypertext Preprocessor***.[8]

PHP code is usually processed on a web server by a PHP interpreter implemented as a module, a daemon or as a Common Gateway Interface (CGI) executable. On a web server, the result of the interpreted and executed PHP code – which may be any type of data, such as generated HTML or binary image data – would form the whole or part of an HTTP response. Various web template systems, web content management systems, and web frameworks exist which can be employed to orchestrate or facilitate the generation of that response. Additionally, PHP can be used for many programming tasks outside the web context, such as standalone graphical applications[9] and robotic drone control.[10] PHP code can also be directly executed from the command line.

The standard PHP interpreter, powered by the Zend Engine, is free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers on a variety of operating systems and platforms.[11]

The PHP language evolved without a written formal specification or standard until 2014, with the original implementation acting as the *de facto* standard which other implementations aimed to follow. Since 2014, work has gone on to create a formal PHP specification.[12]

W3Techs reports that, as of January 2022, "PHP is used by 78.1% of all the websites whose server-side programming language we know."[13] PHP version 7.4 is the most used version. Support for version 7.3 was dropped on 6 December 2021.

Contents

History



Rasmus Lerdorf, creator of PHP; and Andi Gutmans and Zeev Suraski, creators of the Zend Engine

Early history

PHP development began in 1994 when Rasmus Lerdorf wrote several Common Gateway Interface (CGI) programs in C,[14][15] which he used to maintain his personal homepage. He extended them to work with web forms and to communicate with databases, and called this implementation "Personal Home Page/Forms Interpreter" or PHP/FI.

PHP/FI could be used to build simple, dynamic web applications. To accelerate bug reporting and improve the code, Lerdorf initially announced the release of PHP/FI as "Personal Home Page Tools (PHP Tools) version 1.0" on the Usenet discussion group *comp.infosystems.www.authoring.cgi* on June 8, 1995.[1][16] This release already had the basic functionality that PHP has today. This included Perl-like variables, form handling, and the ability to embed HTML. The syntax resembled that of Perl, but was simpler, more limited and less consistent.[7]

An example of the early PHP syntax:[17]

```
<!--include /text/header.html-->

<!--getenv HTTP_USER_AGENT-->
<!--if substr $exec_result Mozilla-->
  Hey, you are using Netscape!<p>
<!--endif-->

<!--sql database select * from table where user='$username'-->
<!--ifless $numentries 1-->
  Sorry, that record does not exist<p>
<!--endif exit-->
  Welcome <!--$user-->!<p>
  You have <!--$index:0--> credits left in your account.<p>

<!--include /text/footer.html-->
```

Early PHP was not intended to be a new programming language, and grew organically, with Lerdorf noting in retrospect: "I don't know how to stop it, there was never any intent to write a programming language [...] I have absolutely no idea how to write a programming language, I just kept adding the next logical step on the way."[18] A development team began to form and, after months of work and beta testing, officially released PHP/FI 2 in November 1997.

The fact that PHP was not originally designed, but instead was developed organically has led to inconsistent naming of functions and inconsistent ordering of their parameters.[19] In some cases, the function names were chosen to match the lower-level libraries which PHP was "wrapping",[20] while in some very early versions of PHP the length of the function names was used internally as a hash function, so names were chosen to improve the distribution of hash values.[21]

PHP 3 and 4

This is an example of PHP code for the WordPress content management system.

Zeev Suraski and Andi Gutmans rewrote the parser in 1997 and formed the base of PHP 3, changing the language's name to the recursive acronym *PHP: Hypertext Preprocessor*.[7][22] Afterwards, public testing of PHP 3 began, and the official launch came in June 1998. Suraski and Gutmans then started a new rewrite of PHP's core, producing the Zend Engine in 1999.[23] They also founded Zend Technologies in Ramat Gan, Israel.[7]

On 22 May 2000, PHP 4, powered by the Zend Engine 1.0, was released.[7] By August 2008, this branch had reached version 4.4.9. PHP 4 is now no longer under development and nor are any security updates planned to be released.[24][25]

PHP 5

On 1 July 2004, PHP 5 was released, powered by the new Zend Engine II.[7] PHP 5 included new features such as improved support for object-oriented programming, the PHP Data Objects (PDO) extension (which defines a lightweight and consistent interface for accessing databases), and numerous performance enhancements.[26] In 2008, PHP 5 became the only stable version under development. Late static binding had been missing from previous versions of PHP, and was added in version 5.3.[27][28]

Many high-profile open-source projects ceased to support PHP 4 in new code from February 5, 2008, because of the GoPHP5 initiative,[29] provided by a consortium of PHP developers promoting the transition from PHP 4 to PHP 5.[30][31]

Over time, PHP interpreters became available on most existing 32-bit and 64-bit operating systems, either by building them from the PHP source code, or by using pre-built binaries.[32] For PHP versions 5.3 and 5.4, the only available Microsoft Windows binary distributions were 32-bit IA-32 builds,[33][34] requiring Windows 32-bit compatibility mode while using Internet Information Services (IIS) on a 64-bit Windows platform. PHP version 5.5 made the 64-bit x86-64 builds available for Microsoft Windows.[35]

Official security support for PHP 5.6 ended on 31 December 2018.[36]

PHP 6 and Unicode

PHP received mixed reviews due to lacking native Unicode support at the core language level.[37][38] In 2005, a project headed by Andrei Zmievski was initiated to bring native Unicode support throughout PHP, by embedding the International Components for Unicode (ICU) library, and representing text strings as UTF-16 internally.[39] Since this would cause

major changes both to the internals of the language and to user code, it was planned to release this as version 6.0 of the language, along with other major features then in development.[40]

However, a shortage of developers who understood the necessary changes, and performance problems arising from conversion to and from UTF-16, which is rarely used in a web context, led to delays in the project.[41] As a result, a PHP 5.3 release was created in 2009, with many non-Unicode features back-ported from PHP 6, notably namespaces. In March 2010, the project in its current form was officially abandoned, and a PHP 5.4 release was prepared containing most remaining non-Unicode features from PHP 6, such as traits and closure re-binding.[42] Initial hopes were that a new plan would be formed for Unicode integration, but by 2014 none had been adopted.

PHP 7

During 2014 and 2015, a new major PHP version was developed, PHP 7. The numbering of this version involved some debate among internal developers.[43] While the PHP 6 Unicode experiment had never been released, several articles and book titles referenced the PHP 6 name, which might have caused confusion if a new release were to reuse the name.[44] After a vote, the name PHP 7 was chosen.[45]

The foundation of PHP 7 is a PHP branch that was originally dubbed *PHP next generation* (*phpng*). It was authored by Dmitry Stogov, Xinchen Hui and Nikita Popov,[46] and aimed to optimize PHP performance by refactoring the Zend Engine while retaining near-complete language compatibility.[47] By 14 July 2014, WordPress-based benchmarks, which served as the main benchmark suite for the phpng project, showed an almost 100% increase in performance. Changes from phpng make it easier to improve performance in future versions, as more compact data structures and other changes are seen as better suited for a successful migration to a just-in-time (JIT) compiler.[48] Because of the significant changes, the reworked Zend Engine was called *Zend Engine 3*, succeeding Zend Engine 2 used in PHP 5.[49]

Because of the major internal changes in phpng, it must receive a new major version number of PHP, rather than a minor PHP 5 release, according to PHP's release process.[50] Major versions of PHP are allowed to break backward-compatibility of code and therefore PHP 7 presented an opportunity for other improvements beyond phpng that require backward-compatibility breaks. In particular, it involved the following changes:

- Many fatal or recoverable-level legacy PHP error mechanisms were replaced with modern object-oriented exceptions.[51]
- The syntax for variable dereferencing was reworked to be internally more consistent and complete, allowing the use of the operators `->`, `[]`, `()`,`{}`, and `::`, with arbitrary meaningful left-side expressions.[52]
- Support for legacy PHP 4-style constructor methods was deprecated.[53]
- The behavior of the `foreach` statement was changed to be more predictable.[54]
- Constructors for the few classes built-in to PHP which returned null upon failure were changed to throw an exception instead, for consistency.[55]
- Several unmaintained or deprecated server application programming interfaces (SAPIs) and extensions were removed from the PHP core, most notably the legacy `mysql` extension.[56]
- The behavior of the `list()` operator was changed to remove support for strings.[57]

- Support was removed for legacy ASP-style delimiters `<%` and `%>` and `<script language="php"> ... </script>`.[58]
- An oversight allowing a switch statement to have multiple `default` clauses was fixed.[59]
- Support for hexadecimal number support in some implicit conversions from strings to number types was removed.[60]
- The left-shift and right-shift operators were changed to behave more consistently across platforms.[61]
- Conversions between floating-point numbers and integers were changed (e.g. infinity changed to convert to zero) and implemented more consistently across platforms.[61][62]

PHP 7 also included new language features. Most notably, it introduced return type declarations for functions[63] which complement the existing parameter type declarations, and support for the scalar types (integer, float, string, and boolean) in parameter and return type declarations.[64]

PHP 8

PHP 8 was released on November 26, 2020. PHP 8 is a major version and has breaking changes from previous versions.[65][66] New features and notable changes include:

Just-in-time compilation

Just-in-time compilation is supported in PHP 8.[67]

PHP 8's JIT compiler can provide substantial performance improvements for some use cases,[68][69] while PHP developer Nikita Popov stated that the performance improvements for mo